The Imagination Machine

APF BASIC
Language Reference Manual

apf
electronics inc.

The Imagination Machine

APF BASIC

This version of the popular BASIC computer language has been specifically designed and written for the APF Imagination Machine! Special features of this language include

- Three octave music system
- High and low resolution color graphics
- Hand control input commands
- Four place fixed decimal arithmetic with 12 digit accuracy

The information in this manual is organized and presented for use by experienced programmers. For an introduction to BASIC programming we suggest BASIC Tutor ™ — an interactive teaching system available for your APF personal computer

3

Study the keyboard carefully. Most of the keyboard is like a standard typewriter with several logical keys. In addition to the special keys the Imagination Machine also has two kinds of control keywords across the top of the keyboard. Use of the special keys is described in the section below.

ESC

When ESC is pressed it appears on the screen

CTRL

The CTRL key allows you to use the control keywords printed above the top row of keys. The two sets of control keywords correspond to the two rows of keys. To print GOTO on the screen hold down the CTRL key and press the 8 key. To print CLOAD on the screen hold down the CTRL key and press the RETURN key.

SHIFT

Since the computer has no lower case letters the SHIFT keys are only used to print the symbols on the upper half of the number keys and wherever shown on other keys.

HERE IS

Is not used in the current version of APF BASIC

LINE FEED

Is not used in the current version of APF BASIC

RETURN

The RETURN key acts like the carriage return on a typewriter. It is also used to tell the computer when you are through entering. When this key is pressed the computer will look at what you have typed and decide what to do with it

RUB OUT

The RUBOUT key is used to back space and erase one character at a time

SHIFT

The REPT repositions a used to make any other key used continuously on the screen as long as you hold down both keys up to the maximum line length of 128 characters

BRK

The BRK (break) key causes your program to stop running and return to user input or control mode. It does not function during an input statement or while saving and loading on tape

HAND CONTROLS

There are two hand controllers attached to the MP 1000. One is marked with a 1 for left and the other with an F for Right. They can be accessed with a key 5 function (P 12)

6

VARIABLES
There are two types of variable number variables or letter variables The maximum number of variables is 29

NUMBER VARIABLES – Number variables can be up to 13 digits long ranging from
−999 999 999 9999 to −999 999 999 9999

LETTER VARIABLES – Letter variables can contain up to 100 characters The maximum length of each letter variable must be defined with a dimension statement.

VARIABLES NAMES – Variable names can be up to 3 characters long but only the first 3 characters are used Names cannot contain a keyword The first character must be a letter Letter variable names must end with a $
EXAMPLE Variable names that are correct but different
A A$ A1 A1$ A2SAM A2SAM$
Variable names that the computer sees as being the same
AL ALSAM ALLEN AL123
Variable names that are illegal and don't work
3AL AM6 A3#

ALGEBRAIC OPERATORS
The symbols used for arithmetic operations are
+ a + b   arithmetic addition   + a plus b
− a − b   arithmetic subtraction a minus b
* a * b   multiplication   a times b
/ a / b   division   a divided by b
↑ a ↑ b   raise to the power   a to the b power—note
                                    b must be an integer

RELATIONAL AND LOGICAL OPERATORS
Used to compare 2 values, produces a true or false result
= a = b   is equal to
< a < b   is less than
> a > b   is greater than
<= a <= b is less than or equal to
>= a >= b is greater than or equal to

RULES OF PRECEDENCE
Expressions are evaluated with operators having the following precedence
highest precedence   negation
                           exponentiation
                           multiplication/division
                           addition/subtraction
lowest precedence   relational
When 2 operators of the same precedence occur the expression is evaluated left to right

LINE LENGTH
Maximum line length is 128 characters Keywords count as one character

SPACING
All spaces are ignored except those inside quotation marks or in REM statements, or in a print using instruction.

MULTIPLE STATEMENTS/LINE
Multiple statements are separated by a colon
ie 10 PRINT 123 IF A = B THEN GOTO 30

MAX LINE #
Maximum line # is 9999 All line #s must be integer only

7

EDIT

The EDIT statement is used to change a portion of a line in your program. It must be followed by the line number you want to change and the ESC key. Then type the section of the statement you want to change. Press the ESC key again and then the new characters.

EXAMPLE: EDIT N ESC X ESC Y RETURN
    N is line N to be edited
    X is to be deleted
    Y is text to replace X

EXAMPLE: 100 INPUT A$ PRINT IN$
    300 IF IN$ = NOT THEN 1000
    300 IF IN$ = YES THEN 1100
    EDIT 100AIN
    EDIT 300T$
    EDIT 300T$
    LIST

    100 INPUT IN PRINT IN$
    300 IF IN$ = NO THEN 1000
    300 IF IN$ = YES THEN RETURN

LIST

Lists all lines in your program

LIST 100

Lists line 100 of program

LIST 100

Lists line from 100 to end of program

LIST 100,5

Lists five lines of your program beginning with line number 100

RUN

Sets all variables to zero. Execution then begins with the first line in your program

CALL

Transfers program execution to a machine language routine written in M 6800 object code.

EXAMPLE: CALL 17040
The call transfers your program to a special sub routine that begins at location 17040 this is machine code and uses the HV section the TV section. When a 6800 RTS instruction occurs return will be to the next basic statement

DATA 5 10 BILL

Stores data to be accessed by READ statements. Data items will be read sequentially items may be numbers or characters but not expressions. Each data item must be separated by a comma

DIM

Reserves memory for storage lists of strings or tables of numbers
EXAMPLE: DIM A$(10,8),K 12),G5,8)
    reserves computer memory for an 11 character string (A$),& strings of 13 characters each (B$) and a table of numbers that is 6 rows by 9 columns (G) Maximum string length is A$(99) Maximum subscript A$(99)

8

END
Stops program and returns control to the console

FOR A = 10 TO C
Combines with a NEXT statement to form a loop. Variable A begins with the value of B and increments to the value of C.

FOR A = 10 TO 1 STEP -1
Combines with a NEXT statement to count from 10 to 1. Note: if step value is not given, it is assumed to be 1.
```
10 FOR A = 10 TO 1 STEP -1
20 PRINT A
30 NEXT A
```

GOSUB 100
Calls a BASIC subroutine located at line 100

GOTO 100
Transfers program to line 100

IF A = 10 THEN 300
Transfers program to line 300 if A equals B. If A is not equal to B, the rest of the line is not read so the computer transfers to the next numbered line. Any comparison can be used, such as

   = equal
   < > not equal
   > greater than
   < less than
   > = greater than or equal
   < = less than or equal

IF A$ = "B$ THEN GOSUB 2000
Transfers program to the subroutine at 2000 if A$ is less than B$. The value of A$ is compared to the value of B$ alphabetically. Numerals come before letters alphabetically.

IF A = B THEN PRINT "EQUAL" END
Prints the word EQUAL and ends the program if A and B are not equal. If A equals B then the rest of the program is executed

INPUT A
Reads data from the keyboard. A question mark is displayed to indicate that the computer is waiting for a response. Single multiple input items with a comma. Command is only terminated after a return key is pressed
EXAMPLE: INPUT A B

The user must respond with two numbers separated by commas. If only one input is given, then computer automatically asks for second one (R)

INPUT A$ B
Reads input where a character string comma and a number

INPUT "AMOUNT", A
Prints the word AMOUNT followed by a question mark and waits for the user to type in a number

NOTE If a string being requested and the user responds with a group of characters beginning with a letter then the computer accepts the input as it if types a it
If the user types a group of characters beginning with one or more numerals the computer accepts those numerals as the number

9

LET A = B or A = B

Sets the variable A equal to the constant B. The word LET is optional and A = also
works. The constant B can be replaced by an arithmetic expression

EXAMPLE A = B + B*C or
B = B/C A = C*C

This example is equivalent to B = (B/C) A/2*C)

ON A GOSUB 1000 1200 1500 2000

Calls one of a list of subroutines, depending on the value of A. If A equals one, GOSUB
1000 results. If A equals two, GOSUB 1200 results and if A equals three, GOSUB
1500 results. If A equals any other value than GOSUB 2000 results

ON A GOTO 10 20 30

Jumps to one of a list of lines, depending on the value of A. If A equals one, GOTO
10 results and A equals two, GOTO 20 results. For any other value results in a GOTO 30

POKE 784 1

Places the number 1 in memory location 784. The POKE places the number 1 in the
middle of your screen. Any memory location can be poked as long as it is between
0 and 65 536. The value poked must be a positive integer from 0 to 255. (If greater
than 255, module 256 is performed

PRINT A

Prints the value of A

PRINT A B

Prints the values of A and B in separate columns. Columns are automatically char-
acter-wide (a. A is column 1   B is printed in column 9)

PRINT A; B

Prints the values of A and B with no spaces between them.

PRINT A TAB(10) B

Prints the value of A, spaces ten characters from the left margin and prints the value
of B

PRINT A SPC(10) B

Prints the value of A, spaces ten characters from the last character of A, and prints
the value of B

PRINT "HELLO"

Prints the word HELLO

PRINT USING "$$### ## " A

Prints the value of A with a leading dollar sign and two decimal places. $29.50
$120.01 $27.49 You must use as many dollar sign marks (#) as there are digits in the
value of A

PRINT USING "########## TOTAL IS ### " A$ B

Replaces the pound signs (#) with the characters in A$, the word TOTAL is SATUR-
DAY TOTAL IS 128 DOOMSDAY TOTAL IS 27 SATURDAY IS 365 etc

PRINT USING A$ B

Prints the value of B using the characters in A$ as the format. The format string is
contained in the string variable A$

10

PRINT USING 100,A

Prints the value of A using the data in line 100 as the format. Line 100 must begin with a colons and quotes are not required

```
100 ## ###
100 $$$### ##
100 LAST QUOTE ####
```

READ A,B\$

Reads data into specified variable from DATA statement

EXAMPLE 100 DATA 37,BILL SAM 8

    200 READ A,B\$

    300 READ C\$,D,E F\$

    300 DATA A2 THE

Note that DATA statements can be placed after the READ statements

RESTORE

Resets the data pointer so that the next READ statement begins with the first DATA statement in the program

EXAMPLE 100 DATA 37,BILL 8 SAM

    200 READ A,B\$

    300 RESTORE

    400 READ C,D,E\$

In the example C will be assigned the value 37

RETURN

Returns from the subroutine to the statement following the GOSUB.

STOP

Stops the program

CHAPTER 5 ARITHMETIC FUNCTIONS

All functions are of the form

    NAME(x)

Name is the function mnemonic. No spaces are allowed between the name and the left parentheses

X is the operand and can be a constant expression or function

ABS(x) —returns absolute value of x

    ABS(3) = 3

    ABS(0) = 0

    ABS(1.45) = 1.45

INT(x)—returns integer portion of x. Fractional portion of x is truncated

    INT(12.34) = 12

PEEK(x)—returns decimal value stored in memory location x. X must be a number

EXAMPLE A = PEEK (41452)

In this example the value of memory location 41452 is placed in A. If it is 0 then CSAVE will place the TV image at the beginning of your tape

RND(X) random number between 0 and 9.9999

SGN(x) 1 if x>0 -1 if x<0 0 if x=0

11

ASC(A$) Returns ASCII Code of the first character in A$
    ex PRINT ASC(A)
          65   This is decimal ASCII code for letter A

CHR$(A) Produces string of one character whose ASCII code is decimal A
    Ex PRINT CHR$(31)
                This is the litreal whose ASCII code is decimal 31

LEN(A$) Returns # of characters (Length) in A$
    Ex DIM A$ (10)
    A$ = ABC
    PRINT LEN (A$)
         3

KEY$(0)Reads the typewriter keyboard If no key is pressed A$= ~ If key is pressed
    A$ = C where C is the character typed
    EXAMPLE 10 A$ = KEY$(0)   IF A$ = ~ THEN 10
           20 PRINT A$
Line 10 is asking if any key on the keyboard is pressed If no it loops back to the
beginning of line 10 and asks again If a key is pressed, the program drops through
to line 20 and prints that key on the screen

A$=KEY$(2)
Reads the left hand control If no key is pressed A$= ~ If key 0 to 9 is pressed
A$ = ~ where n is the number of the key If the knob is moved A$= d where d is
the direction in N E or W if CL is pressed A$= P if either EN or the FIRE button
is pressed A$= ~

A$=KEY$(1)
Reads the right hand control in the same manner as described above

ADDRESSING CHARACTERS IN A STRING
Since strings are dimensional as arrays it is possible to address any character or
group of characters in a string
    EXAMPLE
    DIM A$(10)
    A$ = ABCDEFGHIJ
    PRINT A$ This prints ABCDEFGHIJ
    PRINT A$(4)   This prints EFGHIJ
    B$ = A$(4,5)   This prints DE

Using the MUSIC command you can write and play music on your computer using numbers. The numbers 1 through 7 play the musical scale.

To play one octave higher, place a multiply sign (∗) in front of the number; and to play one octave lower place a divide sign (/) in front of the number.

You can also play sharps and flats by typing a + or − sign in front of the note. The number zero (0) is used to hold the previous note.
Leaving a space between numbers creates a pause in the music.

EXAMPLE MUSIC 123 1333
        MUSIC 1032103030300

If the music does not play the way you want it flow by placing a 9 between each note by using the command
        MUSIC 3 2 1 2 3 3 0

Spaces create a different sound
        MUSIC 3 2 1 2 3 3 0

If you want to use the same sequence of notes over and over you can assign them to a variable

EXAMPLE   MUSIC A,B$
          10 For I=1 to 10
          20 Music 32133333
          30 Next I
          MUSIC 32123332233 23 32233 23 32233 22

EXAMPLE   Using A$,B$
          A$= 5552332
          B$= 77665
          MUSIC A$, B$

The low resolution graphics mode divides the screen in 512 cursor size rectangles 32 columns wide by 16 rows high (32 x 16). These are the same 512 boxes used to show alphanumerics and therefore this mode allows mixing of alphanumerics and color graphics on the screen

COLOR

Any cursor position can be any one of eight colors at any time by assigning it a color number from 0 to 7 Setting COLOR= expression determines the color to be used in the next line or plot command Expression can be a constant or expression Modulo arithmetic is performed so the expression can be greater than 7

```
0   =   Dark Green        4   =   White
1   =   Yellow            5   =   Green
2   =   Red               6   =   Purple
3   =   Blue              7   =   Orange
```

This means you can have eight different colors on the screen at the same time

SHAPE

To get higher resolution each cursor position can be divided into four parts. By selecting the appropriate shapes you can draw a picture with much higher resolution (64 x 32)

The shape table shows the number for each cursor shape.
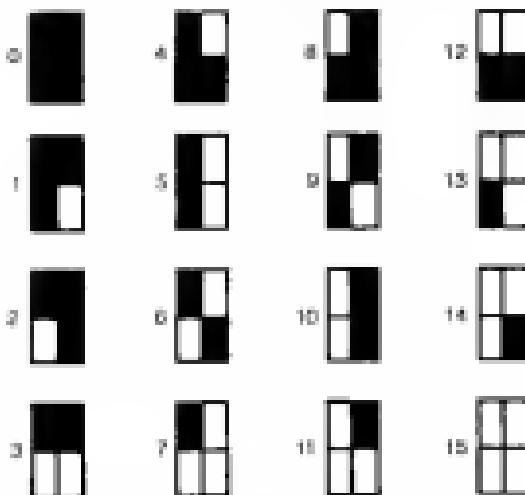
Shape is set equal to a constant or expression it's value is used in the next plot or line command modulo 15 arithmetic is performed so shape can be greater than 15

Since only one color can be assigned to a cursor position the cells that are lit will all be the same color and the remaining cells will be black

```
SHAPE    6     BLACK    YELLOW
COLOR    1     YELLOW    BLACK
```

PLOT

Set the X and Y position on the screen where you want the figure to appear by using the PLOT command

14

PLOT 10.8

Means place the shape in the 10th column on the 8th row
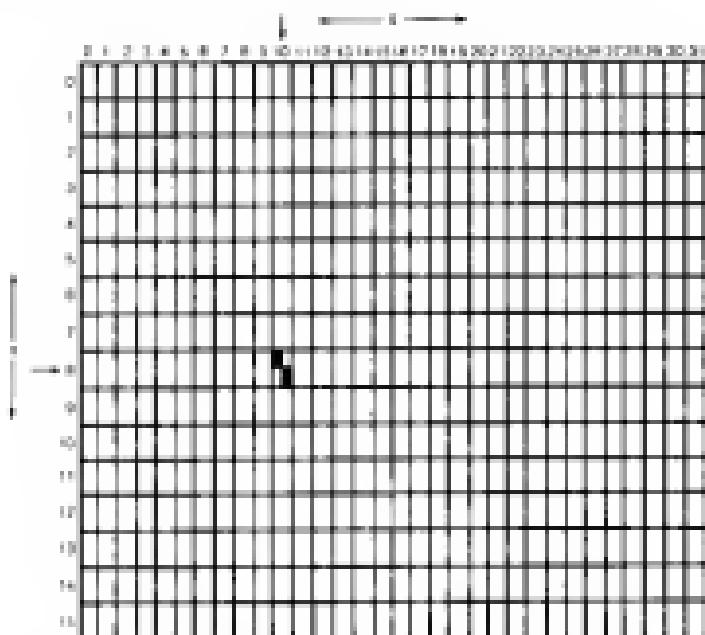
To place shape 6 on the screen in yellow and black in the position X = 10, Y = 8 the command would be

   COLOR = 1 SHAPE = 6 PLOT 10.8

LINE

To draw a horizontal line of the selected shape see the HLIN command and specify the starting column, the ending column and the row

HLIN 10.20.8

To draw a line starting in column 10 ending in column 20 using row 8. The same process is used to draw vertical lines on the screen.

HLIN 10.1, 10-E.1,8-1

Means draw a vertical line start in row 8 easing in row 12 using column 10

NOTE If you plot a shape on top of another shape the second shape is added to the first shape it does not replace it. This new shape takes on the new color. Plotting a 9 on top of a 6 gives you a 15

EXAMPLE 10 SHAPE=15.
   20 FOR I=0 to 7
   30 COLOR=I
   40 HLIN 1,31−I,1
   50 HLIN 1,31−I, 15−I
   60 VLIN I, 16−I, 1
   70 VLIN I, 16−I,31−1
   80 NEXT I
   90 GO TO 90

CLEAR SCREEN

To clear the screen use CALL 17046

15

EXAMINE SCREEN LOCATION

A - PEEK(53)*1*32 + X)

EXAMPLE

PRINT PEEK(512 + 10*32 + 19)

This prints the contents of the 16th position of the 10th row. The numbers 512 and 32 are always used as shown. The other two numbers determine the X and Y coordinate of the screen.

If A<127 the location contains a character (A B C etc)

If A >127 the location contains a character (A B C etc)

0≤A≤63 light green character on dark green

A = □□□□□□□□
    CHARACTER

    CHARACTER CODE (APPENDIX A)

64≤A≤127 dark green character on light green
A = □□□□□□□□ CHARACTER + 64
    CHARACTER CODE (APPENDIX A)

128≤A≤255 shape
A = □□□□□□□□ = 128 + COLOR*16 + SHAPE
    COLOR   SHAPE

REVERSE VIDEO (BLACK/ORANGE)

POKE 8193 50

Changes color of reverse video characters from dark green to light green to black or orange. This does not affect letters in standard video. light green on dark green.

REVERSE VIDEO (GREEN/GREEN)

POKE 8193 57

Restores standard video. Use after POKE 8193 60

LOCATE CURSOR

A Peek (40960/256 + PEEK(40961))
Places in A the position of the cursor

A ROW*32 | COLUMN |
         ROW   COLUMN

POSITION CURSOR

Position cursor by setting A as shown and poking two locations. To position cursor at 22.7

- L=7
- P=22
- A=L*32 + P + 512
- POKE 40960,A/256
- POKE 40961,A-INT(A/256)*256

16

Graphics mode (128 x 192)—There is a high resolution graphics mode which has the following features.
1. 128 x 192 resolution
2. 2 color sets each with 4 colors so there can be a total of 8 colors on the screen at once

MODE SELECT
Enter the graphics mode with
POKE 8193,60
POKE 8194,158
Return to the (64 x 32) graphic mode with
POKE 8193,52
POKE 8194,30
When the computer is in this mode there is no alphanumeric capability. Therefore, pressing keys or the keyboard will not produce letters on the screen but will cause the computer to infer a directed action

MEMORY MAPS
Two memory areas are used. The screen map occupies memory locations 9 to 383 and the shape table occupies locations 512 to 1033

SCREEN MAP
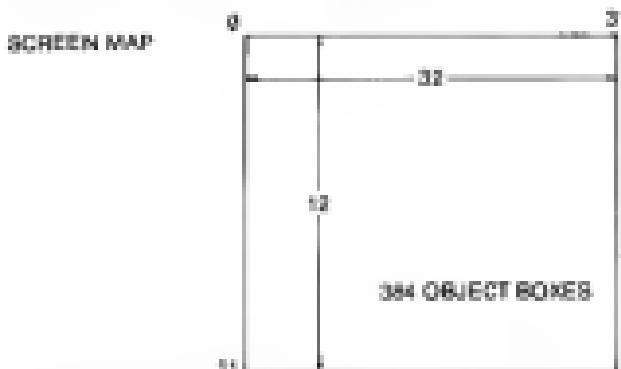
384 OBJECT BOXES

The TV screen is divided into 384 object boxes (12 rows by 32 columns). Each box must occupy any one of 32 special shapes

These shapes are stored in the shape table and numbered 0 to 31
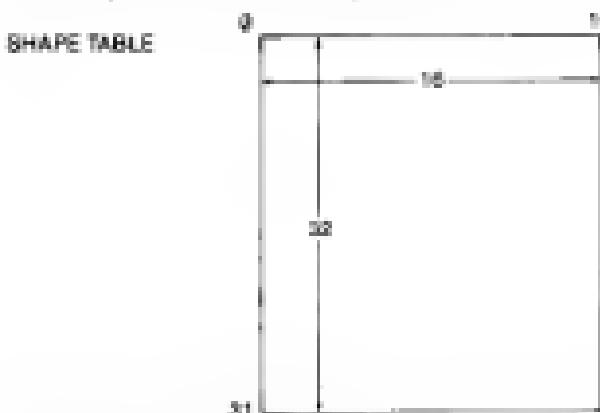
SHAPE TABLE

The shape table contains each of the 32 shapes that can be displayed. Each shape is composed of 16 bytes, one byte for each horizontal row of dots in the shape

DEFINE SHAPES

As in the graphics mode each shape to be plotted on the screen is 8 dots wide and 16 dots high.

Each horizontal row of four dots is set with a POKE instruction. For example, this POKE sets the top row of dots in shape number one to the colors C1, C3, C0, and C4. POKE 512 C1*64+C2*16+C3*4+C4

Each color number is between 0 and 3 depending on the color selected. The color tables are in the next section.

The example at the end of this section shows how up to 32 shapes can be created with similar POKE instructions.

PLOT SHAPES

After the shapes have been created in the shape table they can be plotted on the screen.

The format for each byte in the screen map is

```
□ □ □ □ □
```
       SHAPE NUMBER

If color table select   0

COLOR TABLE

If color table select	0
0	GREEN
1	YELLOW
2	BLUE
3	RED

If color table select	1
0	WHITE
1	AQUA
2	PURPLE
3	ORANGE

NOTE: The color table is selected in the screen map and the specific color in the table is indicated in the shape definition.

EXAMPLE

```
100 L =6 REM LINES ARE 0 TO 11
110 S=12 REM SHAPES ARE 0 TO 31
120 B=12*6 REM COLOR TABLES ARE 0 OR 1
130 DATA 0 Q 1 1 REM A SHAPE IS 4 DOTS ACROSS
140 DATA 0 0 1 1 REM BUILD BY 16 DOTS DOWN
150 DATA 0 0 1 1 2 REM EACH DOT IS ONE OF
160 DATA 0 1 1 2 REM 4 COLORS FROM COLOR TABLE
170 DATA 0 1 2 2 REM COLORS
180 DATA 0 1 2 2 REM EACH COLOR IS NUMBERED 0 TO 3
190 DATA 1 2 2 3
200 DATA 1 2 2 3
210 DATA 2 2 3 3
220 DATA 2 3 3
230 DATA 2 3 3
240 DATA 3 3 3
250 DATA 3 3 3
260 DATA 3 3 3 0
290 DATA 3 0 0 1
300 REM
310 REM SET UP THE SHAPE
320 REM
330 FOR I G TO 15
```

18

```
340 READ A B C D
360 POKE 512 +B*16 I ,A*64+ B*16 I C+1 D
360 NEXT X T
370 REM
380 REM PLACE THE SHAPE ON THE SCREEN USING
390 REM THE CHOSEN COLOR TABLE
400 REM
410 POKE L*32+ P 5 + T*64
```

CHAPTER 10 TAPE

Tape Operation
The tape system is specially designed for the Imagination Machine. The key features are:

1. Dual track tapes are used. One for standard audio recording or playback and the other for digital recording or playback.
2. Microphone jack built in for audio recording. Any standard cassette microphone may be used.
3. Separate speaker built in for audio playback. Volume control is included.
4. Semi-automatic computer control. Record or playback of either audio or digital is under computer control. Fast forward and rewind are manually operated.
5. High speed digital transfer rate — approximately 1600 bits per second.
6. Tape counter included to allow manual location (thru fast forward or rewind) of a desired section of tape

SAVING PROGRAMS
CSAVE is used to save program memory. It is not necessary to specify the record button on the tape deck since that is used only to record audio. Do not delay more than 5 seconds when pressing the play/save button and the return key. For an X4 system CSAVE takes about 45 seconds.

Saving 512 Block for Screen
In addition to the program memory, CSAVE first saves a block of 512 bytes. This will be placed on the screen first when loading so pictures or messages can be stored. There are 2 possible blocks that can be saved. The selected one is determined by the contents of memory location 41452.
If (41452) = 0 then locations 512-1023 (screen memory) are saved
If (41452) ≠ 0 then locations 0-511 (scratch memory) are saved

Saving Program Data
Some CSAVE always saves all memory then all processed variables are also saved on tape. This can be used to save and move data. Loading is again simpler to start the program after loading from an automatically clears all variables.
Note: after a reset is pressed, the RUN command is always required to initialize certain system parameters. Therefore we recommend that if you want to use a tape with data do the following sequence:

```
1 CLOAD
  RUN                              1st RUN after reset
  Follow instructions to load
  When OK appears
  Type GOTO + where + is a first statement of loaded program
```

LOADING TAPES
CLOAD is the command for loading tapes. The first 512 bytes of the tape are always put to screen memory. After that the next 8K bytes are read to program memory. If the end of loading there is a check sum to indicate of data is good correctly if it was the word OK appears on the left side of the screen and the cursor returns. If anything except OK appears it was a bad load and the tape must be reloaded

19

AUDIO PLAYBACK

One of the neat features of the Imagination Machine is that you can record your voice or even music on the same tape as your computer program. To do this, plug a microphone into the microphone jack on the body of the keyboard unit and press the record key. Now type in the following statement

```
POKE 24578,87
```

This turns on the motor. Now talk or play music into the microphone. It is being recorded on the audio track of the tape without damaging or changing any programs you may have placed on the program track.
To turn off the motor type.

```
POKE 24578,38
```

AUDIO SHUTOFF

To stop the voice track on a program tape from playing through the speaker type

```
POKE 24578,54
```

MOTOR START/STOP

To start the cassette tape motor and listen to the audio track type

```
POKE 24578,87
```

To stop the cassette tape motor type

```
POKE 24578,38
```

APPENDIX A

RESERVED WORDS

The following words can not be used as or contained variable names

ABS	MUSIC
ASC	NEXT
CALL	ON
CHR$	OPEN
CLOAD	PEEK
COLOR	PLOT
CSAVE	POKE
DATA	PRINT
DIM	READ
END	RESTORE
FOR	RETURN
GOSUB	RND
GOTO	RUN
IF	SGN
INPUT	SHAPE
INT	SPC
KEYS	STEP
LEN	STOP
LET	TAB
LIST	THEN
	TO
	USING
	VLN

20

APPENDIX B

ASCII CODES AND TOKENS

The following are codes used to place alphanumerics on the screen:

Decimal	Will Appear	Decimal	Will Appear
0	@	32	Space
1	A	33	!
2	B	34	"
3	C	35	#
4	D	36	$
5	E	37	%
6	F	38	&
7	G	39	'
8	H	40	(
9	I	41	)
10	J	42	*
11	K	43	+
12	L	44	,
13	M	45	-
14	N	46	.
15	O	47	/
16	P	48	0
17	Q	49	1
18	R	50	2
19	S	51	3
20	T	52	4
21	U	53	5
22	V	54	6
23	W	55	7
24	X	56	8
25	Y	57	9
26	Z	58	:
27	[	59	;
28	\	60	<
29	]	61	=
30	↑	62	>
31	←	63	?

APPENDIX C

MACHINE LANGUAGE REFERENCE

The iMagination Machine contains a machine language monitor. You can use the monitor to create, display, change, and execute machine language programs. To use the appendix, you must be able to work with programs in 6800 machine language code. You will need a working knowledge of hexadecimal notation.

CALL 28672

This BASIC statement takes you out of BASIC. You are now talking to the iMagination Machine Monitor. The Monitor puts a "*" at the beginning of each line on the screen. When you see the "*", you can enter one of the three monitor commands:

M to examine data in a hexadecimal address
G nnnn where nnnn is a hexadecimal address
G nnnn where nnnn is a hexadecimal address

D nnnn − DISPLAY MEMORY

This command will display the 18 bytes of memory beginning at address nnnn. To display the next 18 bytes, press the "?" key. To end the command, press the RETURN key.

Example D 9B3C
9B3C 20 E0 86 A0 9B 9D 9A 96 CE
A0 9C 7B 9A 26 7C A9 :
9B4C AA 20 24 96 0F CE A0 AA 0C
65 00 00 00 80 00 80 80 FF :

G nnnn − GOTO MEMORY ADDRESS

This command acts much like the BASIC GOTO statement except the value NNNN is a four digit hexadecimal number. The computer immediately begins executing the machine language program in that memory.

Example G 98BA
To execute the program 'Machine's BASIC' This is how you can enter BASIC. If you had a BASIC program in memory when you called the monitor, it should still be there.

M nnnn − MODIFY MEMORY

This command immediately displays the contents at memory address nnnn. You can do one of four things:

reply with the "?" key and the command proceeds by displaying the next page in memory.

reply with "?" key and the command proceeds to display the previous page in memory.

reply the RETURN key and the command is ended.

reply with a two-digit hexadecimal number and the RETURN key and the command stores this new number in the current memory position. Then you can press Return, or ? with the return as above.

The M command cannot change the memory location, it will respond with a "?"

APPENDIX Q

ERROR MESSAGES

The following is a list of error messages. If they occur during a program statement, the statement number as well as the message is displayed.

ARITHMETIC OVERFLOW

The result of a computation is greater than 999999999.9999. This problem can be summarized by using the IF statement to test and limit the sizes of the variables in the equation before doing the computations.

DIMENSION

Somebody is wrong in a Dimension Statement (see to greater than 99, etc.).

DIVISION BY ZERO

The result of dividing by zero is undefined. The computer cannot proceed. This problem can be prevented by checking the value of a divisor with an IF statement before doing the division.

EXPRESSION

The expression is not properly formed. Example as in PRINT PEEK 123 − omit-
theses nor use expand supersam of a function, PRINT 234(32 − missing an open
parenthesis, PRINT 1 + 2 − missing operand.

EXPRESSION MISSING

An expression is expected to be located as a statement but is not found
Example FOR M =

EXPRESSION TOO LONG

You did nothing wrong. All computer languages have limits and you just ran up against one of ours. This will occur if an expression has more than 8 nested brackets or parenthesis.

FOR — NEXT

A NEXT statement was executed when no FOR statement had been executed. Check your program. You have to begin a loop with a FOR statement or the logic of your program is incorrect. Somehow you branched to a NEXT statement without first going through its FOR statement. Fix the logic of your program before continuing.

Example   10 next I
           RUN

IF — THEN

The comparison in the IF statement is wrong. For example, IF A*B THEN 100. Fix the statement before running the program.

ILLEGAL MASK SIZE

The format defined for a PRINT USING statement is incorrect. For example there may be more than 9 "+"s to the left of the decimal place, or more than 4 to the right. Correct the PRINT USING format called a mask before running the program.

ILLEGAL VARIABLE

This is a catchall error message when the computer finds something wrong with a variable. Some examples are

A variable name starts with an illegal symbol — i.e., a number (1A=3). You can't use more than 26 variable names. You used a dimensioned variable name that was not dimensioned.

MEMORY FULL

You ran out of memory. This will usually occur in a DIM statement, if an: then reduce the dimension in a DIM statement or shorten your program.

Memory storage is as follows:

1. A numeric variable takes 7 Bytes. A dimensioned numeric array takes 7 Bytes for each element. (i.e. DIM N (385) takes 7*385 Bytes.)
2. Each line takes 2 Bytes for a line + plus 1 Byte for an end of line symbol.
3. Each element of — PRINT FOR and I take 1 Byte.
4. Actual allowable user storage is 7766 Bytes.

A second reason for Memory Full can be due to your program's continuous execution of a FOR statement with no NEXT statement, or continuous execution of a GOSUB with no RETURN.

GOSUB

A RETURN statement was executed without a GOSUB having been executed. You have a logic error in your program. Make sure you do not use a GOTO or fall into a Subroutine. Fix the problem before running the program.

NO LINE — REFERENCED

You are trying to GOTO a line number which doesn't exist in your program. Correct the statement before running your program.

PRINT DELIMITER

There is an error in your PRINT statement. Check to see that each of the items being printed are separated by a comma or semi-colon. Correct the statement before running the program.

QUOTE MISSING

The right quote or a String Constant is missing. Correct the statement before running the program again.

READ – DATA

Either you have executed a READ statement when there is no DATA statement, or there is an error in the DATA statement. If there is an error in the DATA statement, correct it before running the program again.

If you are executing a READ statement, then either you forgot to include a DATA statement or you have tried to READ more variables than you have data in DATA statements. Make sure you have supplied all of the data in the DATA statements, or that you haven't executed a READ statement more times than you had intended.

WHAT

This is a crackball. It usually means that one of the coms in the line should have been a keyword but wasn't recognized as such. Carefully check the line and correct all errors before running the program again.

> 999999999.9999

You have a constant which has more than four decimal places or which is greater than 999999999.9999 or less than –999999999.9999. You will have to change your program so that numbers outside of this range are unnecessary.